AD-A244 286

'ATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1991	Final 1 Jul 88 - 30 Jun 91

4. TITLE AND SUBTITLE

Distributed Belief Systems

5. FUNDING NUMBERS

(2)

DAAL03-88-K-0087

6. AUTHOR(S)

Jack Minker and Donald Perlis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Maryland
College Park, MD 20742

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

ARO 25870.30-MA

11. SUPPLEMENTARY NOTES

The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes research on the logic of commonsense and nonmonotonic reasoning. The unifying theme in this work is that of the semantical basis for commonsense reasoning, in terms both of algorithms (as in logic programming and deductive databases) and of intelligent agents in a complex world.

Work was focused on two principal themes. They are resource-limited belief systems, and logic programming. The problem, brief outline is as follows: How can an intelligent system deal with incomplete or uncertain information, and yet on average get useful answers? There is a large body of research on this topic, which includes all of non-monotonic reasoning and more besides. The attack on the problem emphazized two special contexts: logic programming, and situated (real-time) reasoning.

14. SUBJECT TERMS

Logic Programming, Commonsense Reasoning, Algorithms, Intelligent Systems

15. NUMBER OF PAGES

22

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

92 1 8 278

ARO 25870.30-EC

Distributed Belief Systems

FINAL REPORT

Jack Minker and Donald Perlis

U. S. ARMY RESEARCH OFFICE

ARO PROPOSAL NUMBER: 25870-MA

CONTRACT OR GRANT NUMBER: DAAL03-88-K-0087

PERIOD COVERED BY REPORT: 1 July 1988 – 30 June 1991

NAME OF INSTITUTION: University of Maryland

TABLE OF CONTENTS

1 STATEMENT OF THE PROBLEM STUDIED

This report describes our research work for ARO grant DAAG03-88-K0087, on the logic of commonsense and nonmonotonic reasoning. This work was a extension of our previous ARO grant in the period from 1985 to 1988. The unifying theme in this work is that of the semantical basis for commonsense reasoning, in terms both of algorithms (as in logic programming and deductive databases) and of intelligent agents in a complex world.

We have focused our work on two principal themes. They are resource-limited belief systems, and logic programming. The problem, brief outline is as follows: How can an intelligent system deal with incomplete or uncertain information, and yet on average get useful answers? There is a large body of research on this topic, which includes all of non-monotonic reasoning and more besides. Our attack on the problem emphasized two special contexts: logic programming, and situated (real-time) reasoning.

2 SUMMARY OF RESULTS

The work we accomplished can be indicated in the following outline; a detailed statement is given in the subsequent text:

- Studies in Logic Programming
- Resource-Limited Belief Systems

2.1 Studies in Logic Programming

2.1.1 Introduction

In this section we outline theoretical results that have been developed in the field of logic programming during the past grant period at the University of Maryland. We focus both on what is called definite logic programming and disjunctive logic programming.

By a definite logic program we mean a set of statements that have the following form:

(1) A :- $B_1, ..., B_n$,

where A and the B_i, $1 \leq i \leq n$ are atomic formulae. There are several forms of the formula that one usually distinguishes. If the right hand side of the formula is empty, and the left hand side is not empty, the formula is referred to as a fact. If the left hand side is not empty and the right hand side is not empty, we refer to the formula as a procedure. The atom A is called the procedure head, and the right hand side is the procedure body. The formula is read as "A if B_1 and B_2 and ... and B_n." If both the left and the right hand side are empty, the formula is referred to as the halt statement. Finally, a query is a statement where the left hand side of the formula is empty

and the right hand side is not empty. The formula where the left hand side is not empty and the right hand side may or may not be empty is referred to as a *program clause.* A set of such program clauses is said to constitute a *definite* logic program.

By a disjunctive logic program we mean a program which contains clauses of the form,

(2) $A_1,...,A_m$:- $B_1,...,B_n$,

where the A_i and B_j, $1 \leq i \leq m, 1 \leq j \leq n$ are atoms. A disjunctive clause is read as, "A_1 or A_2 or ... or A_m if B_1 and B_2 and ... and B_n." In both (1) and (2) variables in predicates are assumed to be universally quantified. We refer to clauses such as in (2) clause as *disjunctive program clauses.* The atoms in the left hand side of the clause are disjuncts, while the atoms in the right hand side are conjuncts. If m=1 the disjunctive program clause reduces to a definite program clause.

We shall also consider clauses of the form (1) or (2), where the B_i may be literals. By a *literal* we mean either an atom or the negation of an atom. A clause of the form (1) or (2) which contains literals in the body is referred to as a *normal or general logic program clause.*

This article is divided into several sections. In each section we describe the contributions made by others and then outline the extensions that have been made to logic programming at the University of Maryland. Our contributions have been to extend the foundations of definite logic programming to disjunctive logic programming. In the section *Definite and Disjunctive Logic Programming*, we describe the theory developed to handle definite and disjunctive programs whose forms have been given above. In the section *Negation in Definite and Disjunctive Logic Programming*, we describe the theory developed to handle negated queries in definite and disjunctive logic programs. In the section *Normal or General Logic Programs* we discuss several topics: *Stratified Logic Programming, Well-Founded and Generalized Well-Founded Logic Programming* and *Generalized Disjunctive Well-Founded Logic Programming.* In the subsection *Stratified Logic Programming*, the theory for general logic programs with no recursion through negative literals in the body of a program clause is described. In the subsection *Well-Founded and Generalized Well-Founded Logic Programming*, we describe extensions to the theory that handles general logic programs that are not stratifiable. In the subsection *Generalized Disjunctive Well-Founded Logic Programming*, we extend the results to disjunctive programs. This permits the full gamut of possible programs to be discussed from a theoretical view. We do not treat theories in which the left hand side of a program clause may contain negated atoms.

1.1.2 Definite and Disjunctive Logic Programming

A question that arises with any programming language is that of the semantics of the program. What does a program written in that language mean, and what programs can be written in the language? In a logic program the fact that one is dealing with a set of logical statements permits one to use concepts from classical logic to define a semantics for a logic program. It is convenient and sufficient to focus on the Herbrand domain of a program to capture the semantics of a logic program. By the *Herbrand domain* we mean the set of all terms formed from the set of all constants in the program and recursively the set of all functions whose arguments are terms. If there are no constants in the program, then an arbitrary constant is added to the domain. Given the Herbrand

domain, one can then consider the *Herbrand Base*, which is the set of all ground predicates that can be constructed from the Herbrand domain. It is sufficient to define a semantics for the program over this domain. We use *Herbrand interpretations*, which are subsets of the Herbrand base to specify the semantics of logic programs. A *Herbrand model* of a logic program is an interpretation that satisfies all clauses in the program.

Definite Logic Programming In their 1976 paper, van Emden and Kowalski [86] defined three different semantics for a definite logic program. These are referred to as model theoretic, proof theoretic (or procedural), and fixpoint (or denotational) semantics. Since we are dealing with logic, a natural semantics is to state that the meaning of a definite logic program is a Herbrand model of the theory. However, this definition is too broad as there may be atoms in the Herbrand model that one would not want to conclude to be true. For example, the program $\{p(a), q(b)\}$ has a Herbrand model that consists of $\{p(a), p(b), q(a), q(b)\}$. It is clear that the program does not state that either $p(b)$ or $q(a)$ are true. Van Emden and Kowalski showed that for definite logic programs, the intersection of all Herbrand models of a program is a unique minimal Herbrand model of the program. It is the least model as it is contained within all models. It represents the least amount of information that can be specified as true. The least Herbrand model will be abbreviated to be M_p.

A second semantics that can be associated with a program is a procedural semantics. Gödel showed that one obtains the same results with proof theory as one does from model theory. van Emden and Kowalski showed that if one uses a proof procedure called *Linear Resolution with Selection Function for Definite Programs (SLD-resolution)*, that the ground atoms that are derivable using SLD from the program, forming the *success set, (success(P))* of the program, are exactly the same as the atoms in the least Herbrand model, M_p.

A third semantics is obtained by defining a mapping, T, from Herbrand interpretations to Herbrand interpretations. As in denotational semantics, if the domain over which the mapping, T, is defined is a complete lattice and the mapping is *continuous*, then the mapping, T, has a least fixpoint (lfp(T)), which is taken to be the meaning of the program. By a *fixpoint* we mean that a mapping, T, satisfies the formula T(I) = I, for some I. The following mapping, defined by van Emden and Kowalski, which maps Herbrand interpretations to Herbrand interpretations, is continuous. Let P be a definite program, and I be a subset of the Herbrand Base of P, then

(3) $T_P(I)$ = {A in HB(P): A :- $B_1, B_2, ..., B_n$ is a ground instance of a clause in P, and $B_1, B_2, ..., B_n$ are in I}.
The major result is that $M_P = lfp(T_P) = success(P)$. That is, the model theoretic, the procedural and fixpoint semantics all give the same meaning to a program: the set of ground atoms that are logical consequences of the program.

Disjunctive Logic Programming In a disjunctive logic program, there is not necessarily a unique minimal Herbrand model. Consider the disjunctive program that consists of a single program clause:

(4) $p(a), p(b)$:-
where a and b are constants. This clause is equivalent to

(5) $p(a) \vee p(b)$
and has two minimal Herbrand models, $\{p(a)\}$, and $\{p(b)\}$. Neither model contains the other. Furthermore, the intersection of the two minimal models is not a model. Hence, disjunctive programs do not share the property of definite logic programs that the intersection of models is a model.

Although there is no unique minimal Herbrand model in a disjunctive logic program, there are a set of minimal Herbrand models that do not contain one another. In 1982 Minker [51] developed a suitable model theoretic semantics for disjunctive logic programs in terms of the minimal models. He defined the meaning of the program to be the set of positive ground clauses that are true in every minimal model.

To obtain the proof theoretic semantics of a disjunctive logic program, the inference system, *Linear Resolution with Selection Function for Indefinite Programs (SLI-resolution)*, developed by Minker and Zanon [58] is used. As the proof theoretic semantics one should take the set of all positive ground clauses that one can derived from the program using SLI resolution. We call this set the *success set, (succ(P)), of P.* The proof theoretic and the model theoretic semantics yield the same results.

To obtain the fixpoint semantics of a program, Minker and Rajasekar have modified the van Emden-Kowalski fixpoint operator. When working with disjunctive programs, it is not possible to map Herbrand interpretations to Herbrand interpretations. The natural mapping with a disjunctive theory is to map a set of positive ground disjuncts to positive ground disjuncts. Minker and Rajasekar [78] therefore defined the *Extended Herbrand Base (EHB$_P$)* to consist of the set of all disjuncts that can be constructed using the Herbrand Base. Subsets of the EHB_P under the partial order subset form a lattice. Subsets of the EHB are referred to as *states.* Minker and Rajasekar defined their fixpoint operator to be:

(6) $T_P(S)$ = {A in EHB_P: A' :- $B_1, B_2, ..., B_n$ is a ground instance of a clause in P, and $B_1 \vee C_1, B_2 \vee C_2, ..., B_n \vee C_n$ are in S, and A is the smallest factor of the clause $A' \vee C_1 \vee ... \vee C_n$, where the $C_i, 1 \leq i \leq n$ and A' are positive clauses.}

The major result in disjunctive logic programming is that the model semantics, the proof semantics and the fixpoint semantics yield the same semantics and capture the set of minimal positive clauses that are logical consequences of the program.

There is a corresponding result with respect to disjunctive logic programs that is similar to the minimal model property. Let a *model state* be defined to be a subset of the EHB_P of a program such that every model of the program is contained in or equal to a model of the set of clauses in the model state, and every minimal model of the model state is a model of the program. Then the intersection of all model states is a model state and it is the minimal model state.

2.1.3 Negation in Definite and Disjunctive Programs

Given a definite or a disjunctive program, the only answers that may be derived are positive. It is not possible to answer a negative query. One would have to permit negative information to be stored with the program. However, adding negative data could overwhelm a system as there is an unlimited amount of negative information that may apply. Furthermore, it may be necessary to invoke a full theorem prover to find answers to queries. We describe several ways in which one may conclude negative information from definite or disjunctive theories without having to add negative data to the program. The theories of negation described below lead to nonmonotonic logics, important for commonsense reasoning. By a nonmonotonic logic is meant one in which the addition of a new fact to a theory may cause previous conclusions to become false.

Negation in Definite Programs Reiter [81] and independently Clark [13] were the first to address negation in definite logic programs and deductive databases. Their results appeared in a book edited by Gallaire and Minker [27] entitled, *Logic and Data Bases.* To answer negated queries, Reiter defined the *closed world assumption (CWA).* According to the CWA, one can assume a negated atom to be true if one cannot prove the atom from the program. Reiter showed that the union of the theory and the negated atoms proved by the CWA is consistent. Clark argues that the way in which one should view negation is that the clauses in a definite logic program supply "if" conditions. What one should do is consider definitions to imply "if-and-only-if" conditions. To do so, one effectively reverses the "if" condition to be an "only-if" condition and then takes the union of this set of clauses with the original program clauses. The union of these two sets, augmented by equality axioms is referred to as the *Clark completion operator*, and written comp(P). The program need no longer be definite when the "only-if" conditions are added to the "if" conditions. However, Clark shows that by using *negation as finite failure (NAF)* on the "if" definitions, then one can conclude the negation of an atom if it fails finitely along every branch of the proof. The proof procedure *SLDNF* is used for this purpose. SLDNF is sound and complete with respect to the comp(P).

Shepherdson [84] showed a relationship between answers found using the CWA and the comp(P) theories of negation. He provides conditions under which they are the same and under which they may differ.

Negation in Disjunctive Theories In his discussion of the CWA in 1978, Reiter showed that the CWA applied to disjunctive theories leads to inconsistencies. Consider the theory $\{p(a) \vee p(b)\}$. Since it is neither possible to prove $p(a)$ nor $p(b)$, by the CWA, one may assume $\neg p(a)$ and $\neg p(b)$. But the union, $\{p(a) \vee p(b), \neg p(a), \neg p(b)\}$, is now inconsistent.

To overcome this problem, Minker [51] defined the *Generalized Closed World Assumption (GCWA).* There are two ways to characterize the GCWA: model theoretic and proof theoretic. In the model theoretic approach an atom $p(a)$ is assumed to be false if it is not true in all minimal models. In the proof theoretic approach, one may assume $p(a)$ to be false if for all positive clauses K, whenever $p(a) \vee K$ can be derived from the program, then K can be derived from the program. Minker showed that these two definitions are equivalent. Computing answers with the GCWA is

computationally difficult as shown by Chomicki and Subrahmanian [12].

As shown by Lobo, Minker and Rajasekar [44] , one can define a completion for a disjunctive program in a manner similar to the Clark completion operator. This leads to a theory of negation called the *Weak Generalized Closed World Assumption (WGCWA)*, which is no more difficult to compute than negation in the completion theory associated with definite programs, and computationally less complex than the GCWA. In the WGCWA, we may assume the negation of an atom $p(a)$ if there is no positive ground clause that can be derived from the program that contains p(a). We refer to the completed theory for disjunctive programs as dcomp(P). The WGCWA was discovered independently by Ross and Topor who call it the *disjunctive database rule (DDR)*. Lobo, Minker and Rajasekar show that the GCWA implies the WGCWA. Both the GCWA and the WGCWA compute answers to negated atoms when they are ground.

2.1.4 Normal or General Logic Programs

Definite programs or disjunctive programs do not allow negated atoms in the right hand side of a program clause. This restricts the expressiveness that one may achieve in writing programs with negation in the body of a rule. As noted earlier, programs with negated atoms in the body of a program clause are referred to as normal or general programs. One can, of course, write an equivalent formula for a normal program which does not contain negated atoms, by moving the negated atom to the head of the program clause to achieve a disjunction of atoms in the head of the clause. This, however, has a different connotation than that intended by a negated atom in the body of a clause. It is intended in the latter case that the negated atom be considered solved by a default rule, such as the CWA or negation as finite failure.

As noted by Apt, Blair and Walker [2] , and also by Van Gelder [87], by Naqvi [61] and by Chandra and Harel [11], problems arise with the intended meaning of a program in some instances. For example, the program $\{p : -\neg q, q : -p\}$ raises questions as to its meaning. We describe alternative ways to handle normal definite programs and the corresponding approach with normal disjunctive programs.

Stratified Normal Programs In a stratified program one allows normal programs which do not permit recursion through negation. In the example given in the previous section, one would not permit the program clauses $\{p : -\neg q, q : -p\}$ to be in the program since there is recursion through negation. That is, we have "p if $\neg q$" and "q if p" which needs recursive application of the two rules to solve p and the application is through the negative literal '$\neg q$' When one excludes these constructs, one can place program clauses into different strata. Stratified programs are a simple generalization of a class of programs introduced in the context of deductive databases by Chandra and Harel.

Apt, Blair and Walker showed that if one has a stratified definite normal program, then one can find a fixpoint for the first stratum and then use this fixpoint as the starting point, find the fixpoint of the next stratum and continue until a fixpoint is obtained for the last stratum. This final fixpoint is taken as the meaning of the program. They show that the fixpoint is a model and

furthermore, is a minimal model of the program. The model semantics achieved is independent of the manner in which the program is stratified. Przymusinski [75] has defined the concept of a *perfect model* and has shown that every stratified logic program has exactly one perfect model. It is identical to the model obtained by Apt, Blair and Walker.

Problems arise in programs where a negated literal has to be evaluated and the literal contains a variable. In this case, the program is said to *flounder*. Chan [10] has defined *constructive negation*, that will find correct answers even in the case of negated literals that contain variables. The underlying idea behind constructive negation is to answer queries using formulas involving only equality predicates.

Stratified Disjunctive Programs Rajasekar and Minker [79] apply the nonmonotonic fixpoint semantics developed by Apt, Blair and Walker to a closure operator T^C to develop a fixpoint theory for stratified disjunctive logic programs. In addition, they develop an iterative definition for negation, called the *Generalized Closed World Assumption for Stratified programs (GCWAS)*, and show that the semantics captures this definition. A model-theoretic semantics is developed for stratified disjunctive programs which is shown to be the least state characterized by the fixpoint semantics that corresponds to a stable-state defined in a manner similar to the stable models of Gelfond and Lifschitz [28]. A weaker semantics is also developed for stratification based on the WGCWA.

Lobo [42] extends the concept of constructive negation, introduced by Chan for stratified logic programs, to apply to stratified disjunctive logic programs. The results include the theories of negation for disjunctive logic programs: the GCWAS and the WGCWAS.

Well-Founded and Generalized Well-Founded Logic Programs There exist programs that are not stratifiable and yet we desire to compute answers to queries over these theories. An example of a program that is not stratified is: $\{p : -a, p : -b, a : -\neg b, b : -\neg a\}$. Van Gelder, Ross and Schlipf [88] define the concept of *well-founded semantics* to handle such programs. Przymusinski [73] presents the ideas of well-founded semantics in terms of a three-valued logic consisting of *true*, *false* and *unknown*. Thus, in the above program, we conclude that p, a, and b are all unknown. That is, we cannot conclude that they are true or false. Van Gelder, Ross and Schlipf develop fixpoint and model theoretic semantics for such programs. Ross [83] and Przymusinski [76] develop procedural semantics. The three different semantics are equivalent.

If one analyzes the above program, another meaning to the program is also possible. In particular, the last two program clauses state that a is true if b is not true, and b is true if a is not true, while the first two clauses state that if a is true then p must be true and if b is true p must be true. Thus, although we may not be able to conclude which of a or b are true, we can surely conclude that p must be true. Baral, Lobo and Minker [6] develop model theoretic, fixpoint and procedural semantics to capture the meaning of programs such as given above. They term this *generalized well-founded semantics (GWFS)*. The fixpoint definition is similar to the definitions of well-founded semantics of Przymusinski and iterates over partial interpretations. At each iteration concepts borrowed from the GCWA are introduced to provide the appropriate semantics. Every

atom proved to be true in the well-founded semantics is also true in the generalized well-founded semantics. However, some additional atoms may be proved true in the GWFS.

Generalized Disjunctive Well-Founded Semantics There have been extensions of the well-founded theory to disjunctive logic programs. Ross [82] developed a *strong well-founded semantics* and Przymusinski [74] developed what is called a *stationary semantics.* This work extends well-founded semantics to disjunctive logic programs.

Baral, Lobo and Minker [3, 7] extend the GWFS to *generalized disjunctive well-founded semantics (GDWFS).* They describe fixpoint, model theoretic and procedural semantics and show that they are equivalent. The semantics achieved is stronger than that of either Ross or Przymusinski. The idea of the fixpoint semantics is an extension of the fixpoint semantics for the GWFS and iterates over a pair of sets, one a set of disjunctions of atoms assumed to be true and the other a pair of conjunctions of atoms assumed to be false. Instead of only assuming atoms to be false, the extension permits conjunctions to be false. It is shown that the extension is no more complex to compute than the GCWA.

2.1.5 Summary

We have described the foundational theory that exists for definite normal logic programs and the extensions that have been made at the University of Maryland to that theory and to disjunctive normal logic programs. The theory of definite and disjunctive logic programs applies equally to deductive databases where one typically assumes that the rules are function-free. A firm foundation now exists both for definite normal and disjunctive normal programs for deductive databases and logic programming.

Although we have developed model theoretic, proof theoretic and fixpoint semantics for disjunctive logic programs, efficient techniques will be required for computing answers to queries in disjunctive deductive databases and logic programs. Some preliminary work has been reported by Grant and Minker [53], Liu and Sunderraman [41], and by Henschen and his students Yahya and Park [89, 32]. However, a great deal of additional work is required.

2.2 Resource-Limited Belief Systems

In the area of resource-limited reasoning, we pursued three related topics. Nonmonotonic reasoning has long been known to involve a form of introspective reasoning. We studied this in two aspects: purely formal methods, and real-time (semi-)formal methods. Our purely formal results are reported in [63] [65] [20]. In the first of these papers we showed that negative introspection is indeed computable in many cases, allowing not only for default reasoning but also a more general kind of reasoning about possibilities (i.e., about one's ignorance). In the second paper we showed that certain formalisms for default reasoning are inherently inconsistent, and that a new approach is needed if we are to maintain highly expressive representation languages. In the third paper, we

found a way to circumvent certain difficulties of non-monotonic reasoning in which default conclusions were too liberally drawn, producing intuitively incorrect results; our new approach involved the notion of limiting the "scope" (or range) of application of default reasoning (to objects of immediate interest). All of this involves a weak sense of resource-limitation, in that all that we required was semi-decidability. However, this was already a severe restriction of the much more frequent undecidable formalisms in the literature.

Our second effort in this area involved further work in step-logics, in which real-time methods are employed. Here a formal logic is altered so that the time taken to use the syntactic rules in forming proofs is itself a parameter internal to the well-formed formulas in the logic, and thus as proofs proceed these parameters change their values. The best analogy for such a parameter is simply a watch or clock that the reasoner (computer) looks at as it carries out its reasoning. However, the data derived from the clock (the present time) is used in the course of reasoning. This is particularly useful in deadline situations, since the reasoning must come to a useful conclusion before the deadline is reached. We were able to develop such real-time formalisms sufficiently to be able to solve several classical problems with them, including the Brother Problem of R. Moore (see [17]), the Three-Wise-Men Problem (see [14]), and the Nell & Dudley Problem of McDermott (see [39]), all restated in more realistic temporal terms.

In [40], we showed that circumscriptive methods lend themselves well to reasoning about the ignorance of others, which is important in communication as well as many other multi-agent interactions. Gal and Minker [26] studied the problem of providing cooperative answers to provide better communication between man and machine.

2.2.1 Introspection and Nonmonotonic Reasoning

Nonmonotonic forms of reasoning depend on a means for determining when certain propositions are *not* known. This is so since the usual form of nonmonotonic inference in commonsense reasoning is

$$\text{B \& Consistent(A)} \rightarrow \text{C.}$$

In fact, typically, C is simply A, as in

$$\text{SovietTank(x) \& Consistent(Functional(x))} \rightarrow \text{Functional(x).}$$

Now, consistency of a wff A simply amounts to unprovability of $\neg A$. Thus to reason nonmonotonically as above, an agent must be able to determine that it is unable to deduce $\neg A$, i.e., that it does not have information allowing the conclusion $\neg A$. This facility is called negative introspection.

This presents a problem, since in general the set of logical consequences of given axioms is undecidable. However, we developed a variant of circumscription, called *autocircumscription* that indicates certain cases in which an agent can indeed compute that it does not know (cannot deduce) a wff such as $\neg A$ (see [63]). We applied this successfully to a number of problems in commonsense

reasoning, reported in [40]. Later, Vladimir Lifschitz developed a yet stronger form of autocircumscription that he calls introspective circumscription.

One issue that arises, is that of possible conflicts that can arise when nonmonotonic (default) conclusions are drawn. For if, on the basis of *not* knowing A, it is found that A, then the agent does know A after all. Of course, this is simply a result of temporally distinct states: at first A was not known, and then later it was known. But this presents problems for the formalization of default reasoning which may best be handled by passage to a different form of logic. We have investigated this matter in more detail, indicated in the next section.

2.2.2 Step-Logic

We extended our work on step-logic further into the first-order (non-propositional) case and trained it on some difficult problems in knowledge representation, such as the Three-Wise-Men Problem. We present a variation of this classic problem which was first introduced by McCarthy. A king wishes to know whether his three advisors are as wise as they claim to be. Three chairs are lined up, all facing the same direction, with one behind the other. The wise men are instructed to sit down. The wise man in the back (A) can see the backs of the other two men. The man in the middle (B) can only see the one wise man in front of him (C); and the wise man in front (C) can see neither A nor B. The king begins with five cards, three white, and two black. He throws two (of unknown color) away, then places one card, face up, behind each of the three wise men. The men are given 30 minutes to determine the color of the card that sits behind his own chair. The room is silent; then, after 29 minutes, C says 'My card is white!'.

The reasoning that supposedly occurred is as follows. Because the king started out with three white and two black cards, then threw two away, each wise man must realize there is at least one white card. If the cards of B and C were black, then A would have been able to announce immediately that his card was white. They all eventually come to realize this (they are all truly wise). Since A kept silent past the point at which the others would expect A to have realized this, either B's card is white, or C's is. Shortly after this B should be able to predict, if C's were black, that his card was white. C expects then that B has realized this *by now*. Since B also remains silent, C concludes his card must be white. Clearly each wise man must be able to reason about the time-bound reasoning of the others. We developed a formalism which is capable of performing the above reasoning. See [17].

While this is rather complex reasoning, there are simple counterparts in everyday reasoning. For a military application of this kind of reasoning, consider General A who must decide if the opposing General B has yet broken A's secret code. If B had figured it out, then (so A reasons) B would have reacted to A's messages concerning a plan to build up forces to the South. Since no reaction is observed, A concludes that B has not yet broken the code.

Parallelism comes into this problem in at least two ways: (1) A wise man carries out reasoning in a step-like fashion, but each step may involve several simultaneous deductions, and (2) the three wise men reason concurrently so that a parallel implementation would allow for three copies of a "Wise Man" program to interact with one another, providing experimental material in multiple

agent reasoning. This latter feature ties in with later proposed work below.

We briefly mention a further example in which the effort or time spent is crucial. McDermott [47] and Haas [31] have studied aspects of the problem of saving someone ("little Nell") from an onrushing train. Step-logic should provide a natural context for dealing with the particular difficulty arising from the fact that it is not appropriate to spend hours figuring out a plan to save Nell here, for she will no longer need saving by then. That is, it is often important to take into account the very passage of time as one's reasoning goes on. In part, these are problems of modeling time, as has been studied by Allen [1] and McDermott [47]. However, there is more to it than this. Not only must the agent be able to reason about time, it must be able to reason *in* time. That is, as it makes more deductions, time passes, and this fact itself must be recognized. Otherwise we again face the prospect of losing Nell while deducing that it will take too long to get to a phone to call the train depot. We may even take too long to deduce that it will take too long! Treatments of time in the literature are themselves still in the standard mold of unlimited reasoning, while realistic situations require timely decisions. We have now some results on this problem, reported in [39].

2.2.3 Interagent understanding

Kraus and Perlis [40] showed that one agent can at times reason successfully about another agent's knowledge and lack of knowledge. Although this is not in itself a matter of communication or understanding between agents, it is a necessary preamble to such. For if one agent, A, cannot assess whether another agent, B, does or does not know a certain fact, F, then A's task of communicating with B is made all the harder. If B does know F then A can assume this knowledge on B's part, but otherwise A must first acquaint B with the fact F. The method we used involved the autocircumscription technique of [63], in which the possibility of some assertion, instead of a default about it, is concluded. We were able to apply this to a problem posed by McCarthy: How can one formally conclude that Mikhail Gorbachev does not (as a reasonable assumption) know whether Ronald Reagan is now standing or sitting? [Or, in more military terms, how can we conclude that General Orlov does not (as a reasonable assumption) know whether our tanks are runnning out of fuel?] We were able to solve such problems, in simple cases, by applying autocircumscription to our own knowledge about things that were likely to be sources of knowledge for others. If we could show that we were ignorant of anything that could lead to another's knowledge of fact F, then we were in a reasonable position to assume F was unknown to that person.

2.2.4 Cooperative Answering and Communicating Agents

Gal and Minker have in their papers [20, 24] pursued natural language aspects to the query-answer issue. They have developed a general theory as to how to provide intelligent or cooperative (rather than simply literal) natural language responses to queries. Their approach takes advantage of integrity constraints that exist for a database. Detailed heuristics have been developed for this purpose. A prototype of the method has been implemented in PROLOG. A meta-interpreter written in PROLOG [43] combines integrity constraints with axioms in logic programs. The meta-

interpreter is currently running and has been extended to incorporate some heuristics and aspects of the theory to permit cooperative answers to be obtained, and to produce natural language *output.*

As an example, consider a database with the integrity constraint, "Only senior engineers may earn more than their managers." Then the following query, "List all employees who earn more than their managers," might reasonably be answered by

(i) Only senior engineers may earn more than their managers;
these employees are Smith, Brown, and White.

Note that this avoids both extremes, of ignoring helpful information, as in

(ii) Smith, Brown, and White.

and of presenting irrelevant detail as in

(iii) Smith, Brown, and White are the only such employees; and they are senior engineers all over 50 years old and Smith earns more than Brown or White.

The proper use of heuristics should allow cooperative answers such as (i) while avoiding the others.

A natural language database interface was developed that has four stages:

- A parser translates a natural language query into a logic query.
- An optimizer uses integrity constraints to transform the logic query into an equivalent but more efficient form. Moreover the deductive part of the database is used to express the query with extensional terms only. This stage also collects all information it can gather about integrity constraints related to the question, without searching in the extensional database.
- A response generator receives the optimized logic query and searches in the extensional database for the answers to the query, for the validity of the presuppositions of the query and evaluates those integrity constraints added to the query, which need extensional database search to be evaluated. It then applies selection rules to produce a cooperative response in logic.
- A synthesizer translates the logic response into a natural language response.

Some progress has already been made on each of the four stages. More work on generalizing the heuristics, implementing a meta-interpreter, and developing a natural language paragraph of the answer, is needed. With regard to the latter problem, anaphoric (pronominal) reference is one key issue. For instance, if the optimized logic query leads to the cooperative logic response to the effect that Jack saw Jack in the mirror, e.g., Saw-in-mirror(Jack,Jack) we would want a natural language response such as Jack saw himself in the mirror. Here the pronoun *himself* is substituted

for the second reference to Jack. This is the *inverse* problem of pronominal reference; the direct problem is that of inferring from the second version that *himself* refers to Jack. The latter is the subject of work we undertook for the pronoun 'I' in [49, 48], and which we plan to extend to other pronouns as well.

Directly related to our above work on cooperation principles and natural language is the problem of communicating agents that may share some, but not all, data and syntax. This is a resource-limitation in that each agent has limited access to data and syntax. As an example, we consider agent E and agent F, who speak English and French, respectively. Communication proceeds by a suitable translation function EF from English to French, and FE from French to English. However, not all expressions understood by E may, under EF, produce expressions understood by F, and vice versa. In effect, EF and FE may be partial functions. Moreover, FE and EF may also be multi-valued in that several French expressions may serve as translations of one English expression, and vice versa. Finally, beliefs of E when translated by EF, may not be beliefs of F even when understood by F. Numerous questions arise here, highly relevant to deductive databases and logic programming.

Now, genuine translation algorithms between natural languages are notoriously difficult to construct; indeed so far no one has succeeded in so doing. However, the underlying concept applies to other, perhaps simpler, domains. As an example, suppose E is an (English-speaking) economist and F an (English-speaking) general. E and F may need to communicate about certain logistical issues involving both budgetary aspects and order-of-battle aspects. Although they share a common language, it is a partial sharing in that each knows terminology and concepts unknown to the other. The same broad questions noted above arise here, concerning the ability of E and F to communicate, both in terms of their terminology and also their beliefs.

One way to approach these problems is to view E and F as comprising two large databases written in the same language but in which certain predicates are missing from each. We may suppose given a translation function, or we may try to construct one based on the axioms and facts in E and F. Each of these leads to interesting questions.

For example, it seems likely that a sufficiently massive database of commonsense information, will allow a unique characterization of many commonsense concepts. Thus, the word 'weapon' may, in the context of ordinary information about weapons, be sufficient to distinguish the meaning of that word from, say, 'device that can cause harm' (which is too broad) or from, say, 'deadly device' (which is both too narrow and too broad). We hope to be able to determine conditions (necessary or sufficient or both) related to such unicity. This would aid on the understanding of the nature of communication, since without such a sufficiently massive database to guarantee (nearly) unique characterizations, communication will instead become mis-communication.

Related issues are those undertaken in our study [70, 71] of the nature of reference and meaning in general. That preliminary effort will be developed further as well, especially in the context of communicating agents.

3 RESEARCH PAPERS WRITTEN WITH ARO SUPPORT

During the three year grant from July 1988 to June 1991, we published:

- 14 refereed journal articles
- 28 refereed conference papers
- 15 refereed book chapters

These papers are denoted by an asterisk (*) in the following list.

References

[1] J. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[2] K.R. Apt, H.A. Blair, and A. Walker. Towards a Theory of Declarative Knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann Pub., Washington, D.C., 1988.

[3] C. Baral, J. Lobo, and J. Minker. WF^3: A Semantics for Negation in Normal Disjunctive Logic Programs with Equivalent Proof Methods. manuscript.

* [4] C. Baral, J. Lobo, and J. Minker. Generalized Disjunctive Well-founded Semantics for Logic Programs: Procedural Semantics. In *Methodologies for Intelligent Systems*, volume 5, 1990.

* [5] C. Baral, J. Lobo, and J. Minker. Generalized Disjunctive Well-founded Semantics for Logic Programs: Declarative Semantics. In *Methodologies for Intelligent Systems*, volume 5, 1990.

* [6] C. Baral, J. Lobo, and J. Minker. Generalized Well-Founded Semantics for Logic Programs. In *10th International Conference on Automated Deduction*, West Germany, 1990.

[7] C. Baral and J. Minker. A Constructive Iterative Fixpoint Semantics for Negation in Normal Disjunctive Logic Programs. Under preparation.

* [8] Chitta Baral, Sarit Kraus, and Jack Minker. Combining Multiple Knowledge Bases. *IEEE Transactions*, 1990.

* [9] Chitta Baral, Sarit Kraus, and Jack Minker. Communicating between Multiple Knowledge Based Systems with Different Languages. In *Proceedings of the International Working Conference on Cooperative Knowledge Based Systems*, pages 121–125, University of Keele, England, October 3-5, 1990.

[10] D. Chan. Constructive Negation Based on the Completed Databases. In R.A. Kowalski and K.A. Bowen, editors, *Proc. 5^{th} International Conference and Symposium on Logic Programming*, pages 111–125, Seattle, Washington, August 15-19, 1988.

[11] A. Chandra and D. Harel. Horn clause queries and generalizations. *Journal of Logic Programming*, 2(1):1–15, April 1985.

[12] J. Chomicki and V.S. Subrahmanian. Generalized Closed World Assumption is π_2^0 Complete. Technical Report TR-89-036, Computer Science Department, University of North Carolina, 1989. Also to appera in Information Processing Letters.

[13] K. L. Clark. Negation as Failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

*[14] J. Elgot-Drapkin. *Step-logic: Reasoning Situated in Time.* PhD thesis, Department of Computer Science, University of Maryland, College Park, Maryland, 1988.

*[15] J. Elgot-Drapkin, M. Miller, and D. Perlis. Stop the World—I Want to Think. *International Journal of Intelligent Systems*, 1990.

*[16] J. Elgot-Drapkin, M. Miller, and D. Perlis. *Memory, reason, and time: the step-logic approach.* MIT Press, 1991.

*[17] J. Elgot-Drapkin and D. Perlis. Reasoning Situated in Time I: Basic Concepts. *Journal of Experimental and Theoretical Artificial Intelligence*, 2(1):75–98, 1990.

*[18] D. Etherington, S. Kraus, and D. Perlis. Nonmonotonicity and the Scope of Reasoning. In *Proceedings of the 1st International Workshop on Human and Machine Cognition*, 1989.

*[19] D. Etherington, S. Kraus, and D. Perlis. Limited Scope and Circumscriptive Reasoning. In *Advances in Human and Machine Cognition, Volume 1: The Frame Problem in Artificial Intelligence.* JAI Press, 1990.

*[20] D. Etherington, S. Kraus, and D. Perlis. Nonmonotonicity and the scope of reasoning: Preliminary report. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 600–607, Boston, MA, 1990. AAAI.

*[21] José Alberto Fernández and Jack Minker. Bottom-Up Evaluation of Hierarchical Disjunctive Deductive Databases. In Koichi Furukawa, editor, *Logic Programming: Proceedings of the Eighth International Conference*, pages 660–675, Paris, France, June 1991. MIT Press.

*[22] T. Gaasterland, J. Minker, and A. Rajasekar. Deductive Database Systems and Knowledge Base Systems. In *In Proceedings of VIA'90*, Barcelona, 1990.

*[23] T. Gaasterland, J. Minker, and A. Rajasekar. Knowledge Base Systems A Deductive Database Approach. In *AAAI Workshop on Knowledge Base Management Systems*, Boston, MA, 1990.

*[24] A. Gal. *Cooperative Responses in Deductive Databases.* PhD thesis, University of Maryland, 1988.

*[25] A. Gal and J. Minker. Producing Cooperative Answers in Deductive Databases. In P. St. Dizier and S. Szpakowicz, editors, *Logic and Logic Grammer for Language Processing*, pages 223–254. 1990.

[26] A. Gal and J. Minker. . A natural language database interface that provides cooperative answers. *Proc. Second Conference on Artificial Intelligence Applications*, December 11-13, 1985.

[27] H. Gallaire and J. Minker, editors. *Logic and Databases.* Plenum Press, New York, April 1978.

[28] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R.A. Kowalski and K.A. Bowen, editors, *Proc. 5th International Conference and Symposium on Logic Programming*, pages 1070–1080, Seattle, Washington, August 15-19 1988.

*[29] D. Gordon and D. Perlis. Explicitly biased generalization. *Computational Intelligence*, 5:67–81, 1989.

*[30] J. Grant and J. Minker. Integrity Constraints in Knowledge Based Systems. *Knowledge Engineering*, 1 and 2, 1990.

[31] A. Haas. Possible events, actual events, and robots. *Computational Intelligence*, 1(2):59–70, 1985.

[32] L.J. Henschen and H. Park. Compiling the GCWA in Indefinite Databases. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 395–438. Morgan Kaufmann Pub., Washington, D.C., 1988.

*[33] J.F. Horty. A Skeptical Theory of Mixed Inheritance. In J.M. Dunn and A. Gupta, editors, *Truth or Consequences*, 1990.

*[34] J.F. Horty. Boolean Extensions of Inheritance Networks. In *Proceedings of AAAI-90*, Boston, MA, 1990.

*[35] J.F. Horty. A Credulous Theory of Mixed Inheritance. *Inheritance Hierarchies in Knowledge Representation and Programming Languages*, 1991.

*[36] J.F. Horty. Some Direct Theories of Nonmonotonic Inheritance. In D. Gabbay and C. Hogger, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming.* 1991.

*[37] J.Lobo, J. Minker, and A. Rajasekar. Weak completion theory for non-Horn Programs. In *International Conference and Symposium on Logic Programming*, 1988.

*[38] S. Kraus, M. Nirkhe, and P. Perlis. Deadline-Coupled Real-time Planning. In *Proceedings of 1990 DARPA workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 100–108, San Diego, CA, 1990.

*[39] S. Kraus, M. Nirkhe, and P. Perlis. Planning and Acting in Deadline Situations. In *Proceedings of AAAI-90 Workshop on Planning in Complex Domains*, 1990.

*[40] S. Kraus and D. Perlis. Assessing Others' Knowledge and Ignorance. In *4th International Symposium on Methodologies for Intelligent Systems.* Elsevier Science Publishing Company, Incorporated, 1989.

[41] K.C. Liu and R. Sunderraman. Indefinite and Maybe Information in Relational Databases. To appear in ACM Transactions on Database Systems.

*[42] J. Lobo. On Constructive Negation for Disjunctive Logic Programs. In S. Debray and M. Hermenegildo, editors, *Proc. North American Conference on Logic Programming*, pages 704–718, Austin, Texas, 1990. MIT Press.

*[43] J. Lobo and J. Minker. A metaprogramming approach to semantically optimize queries in deductive databases. In L. Kerschberg, editor, *Proceedings of The Second International Conference on Expert Database Systems*, pages 387–420, Tysons Corner, Virginia, April 1988.

*[44] J. Lobo, J. Minker, and A. Rajasekar. Weak Generalized Closed World Assumption. *Journal of Automated Reasoning, Kluwer Academic Publishers, Netherlands*, 5:293–307, 1989.

*[45] J. Lobo, J. Minker, and A. Rajasekar. On Generalized Disjunctive Logic Programs. In Z.W. Ras and M. Zemankova, editors, *Intelligent Systems: State of the Art and Future Directions.* Ellis Horwood, 1990.

*[46] J. Lobo, J. Minker, and A. Rajasekar. Extending the Semantics of Logic Programs to Disjunctive Logic Programs. In G. Levi and M. Martelli, editors, *Proceedings of the 6th International Conference and Symposium on Logic Programming*, Lisbon, Portugal, June 19-23, 1989.

[47] D. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.

*[48] M. Miller and D. Perlis. Proving facts about 'I'. In *Proceedings of the 10th Int'l Joint Conference on Artificial Intelligence*, pages 499–501, 1987.

*[49] M. Miller and D. Perlis. Proving self-utterances. *Journal of Automated Reasoning*, 3:329–338, 1987.

*[50] M. Miller and D. Perlis. Typicality Constants and Range Defaults: Some Pros and Cons of a Cognitive Model of Default Reasoning. In *Proceedings of the 1991 SIGART International Symposium on Methodologies for Intelligent Systems*, 1991. To appear.

[51] J. Minker. On Indefinite Databases and the Closed World Assumption. In *Lecture Notes in Computer Science 138*, pages 292–308. Springer-Verlag, 1982.

*[52] J. Minker. Toward A Foundation of Disjunctive Logic Programming. In *Proc. of the North American Conference on Logic Programming*, pages 1215–1235, 1989.

*[53] J. Minker and J. Grant. Answering queries in indefinite databases and the null value problem. In P. Kanellakis, editor, *Advances in Computing Research*, pages 247–267. 1986.

*[54] J. Minker, J. Lobo, and A. Rajasekar. Circumscription and Disjunctive Logic Programming. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation.* Academic Press, 1991. To appear.

*[55] J. Minker and A. Rajasekar. Procedural Interpretation of non-Horn logic Programs. In *CADE-9*, 1988.

*[56] J. Minker and A. Rajasekar. Disjunctive Logic Programming. In *Proceedings of the 1989 SIGART International Symposium on Methodologies for Intelligent Systems*, 1989. (Invited Talk).

*[57] J. Minker, A. Rajasekar, and J. Lobo. Theory of Disjunctive Logic Programs. In Jean-Louis Lassez, editor, *Computational Logic: Essays in Honor of J. Alan Robinson*, chapter 19, pages 613–639. MIT Press, Cambridge, Massachusetts, 1991.

[58] J. Minker and G. Zanon. An Extension to Linear Resolution with Selection Function. *Information Processing Letters*, 14(3):191–194, June 1982.

*[59] Jack Minker. An Overview of Nonmonotonic Reasoning. In Anil Nerode, Wiktor Marek, and V.S. Subrahmanian, editors, *The First International Workshop on Logic Programming and Non-monotonic Reasoning*, June 1991. Banquet Address. (Paper in preparation).

*[60] Jack Minker, Arcot Rajasekar, and Jorge Lobo. *Foundations of Disjunctive Logic Programming.* M.I.T. Press, To appear in 1992.

[61] S.A. Naqvi. A Logic for Negation in Database Systems. In J. Minker, editor, *Proc. Workshop on Foundations of Deductive Databases and Logic Programming*, pages 378–387, Washington, D.C., August 18-22, 1986.

*[62] M. Nirkhe, S. Kraus, and P. Perlis. Fully Deadline-Coupled Planning: One Step at a Time. In *Proceedings of 1991 International Symposium on Methodologies for Intelligent Systems*, 1991. To appear.

*[63] D. Perlis. Autocircumscription. *Artificial Intelligence*, 36:223–236, 1988.

*[64] D. Perlis. Commonsense Set Theory. In P. Maes and D. Nardi, editors, *Meta-Level Architectures and Reflection.* North Holland, 1988.

*[65] D. Perlis. Languages with Self Reference II: Knowledge, Belief, and Modality. *Artificial Intelligence*, 34:179–212, 1988.

*[66] D. Perlis. Meta in Logic. In P. Maes and D. Nardi, editors, *Meta-Level Architectures and Reflection*, pages 37–49. Elsevier Science Publishing Company, Incorporated, 1988.

*[67] D. Perlis. Intentionality and defaults. In *Proceedings of the 1st International Workshop on Human and Machine Cognition*, 1989.

*[68] D. Perlis. Truth and Meaning. *Artificial Intelligence*, 39:245–250, 1989.

*[69] D. Perlis. Thing and Thought. In R. Loui H. Kyburg and G. Carlson, editors, *Knowledge Representation and Defeasible Reasoning.* Kluwer, 1990.

*[70] D. Perlis. Putting One's Foot in One's Head - Part I: Why. *Noûs: Special issue on Artificial Intelligence and Cognitive Science*, 1991. To appear.

*[71] D. Perlis. Putting One's Foot in One's Head - Part II: How. In J. Harper and A. Ramsay, editors, *Propositions and attitudes: Between Philosophical logic and Artificial Intelligence.* Kluwer Studies in Linguistics and Philosophy, 1991. To appear.

*[72] D. Perlis, D. Etherington, and S. Kraus. Nonmonotonicity and the scope of reasoning. *Artificial Intelligence*, 1991.

[73] T. Przymusinski. Three-valued Formalizations of Non-monotonic Reasoning and Logic programming. In *Proceedings of First International Conference on Knowledge Representation and Reasoning*, pages 341–348, 1989.

[74] T. Przymusinski. Stationary semantics for disjunctive logic programs and deductive databases. In S. Debray and M. Hermenegildo, editors, *Proc. of the North American Conference on Logic Programming*, Austin, Texas, Oct. 1990.

[75] T.C. Przymusinski. On the declarative semantics of deductive databases and logic programming. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann Pub., Washington, D.C., 1988.

[76] T.C. Przymusinski. Every Logic Program has a Natural Stratification and an Iterated Fixed Point Model. In *"Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principle of Database Systems"*, pages 11–21, 1989.

*[77] A. Rajasekar, J. Lobo, and J. Minker. Skeptical Reasoning and Disjunctive Programs. In *Proceedings of First International Conference on Knowledge Representation and Reasoning*, pages 349–357. Morgan-Kaufmann, 1989.

*[78] A. Rajasekar and J. Minker. A Fixpoint Semantics for Non-Horn Logic Programs. *The Journal of Logic Programming*, 1989.

*[79] A. Rajasekar and J. Minker. Stratification Semantics for General Disjunctive Programs. In E.L. Lusk and R.A. Overbeek, editors, *Proc. North American Conference on Logic Programming*, pages 573–586, 1989.

*[80] A. Rajasekar and J. Minker. On Stratified Disjunctive Programs. *Annals of Mathematics and Artificial Intelligence*, 1, 1990.

[81] R. Reiter. Deductive question-answering on relational data bases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 149–177. Plenum Press, New York, 1978.

[82] K. Ross. Well-founded semantics for disjunctive logic programs. In *Proc. of the first International Conference on Deductive and Object Oriented Databases*, Kyoto, Japan, Dec. 4-6,1989.

[83] K. Ross. A procedural semantics for well founded negation in logic programs. In *"Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principle of Database Systems"*, Philadelphia, Pennsylvania, March, 29-31, 1989.

[84] J.C. Shepherdson. Negation as Finite Failure: A Comparison of Clark's Completed Database and Reiter's Closed World Assumption. *J. Logic Programming*, 1(15):51–79, 1984 June 1984.

*[85] V.S. Subrahmanian and J. Minker. Completion Semantics for General and Disjunctive Logic Programs. In *Methodologies for Intelligent Systems*, volume 5, 1990.

[86] M.H. van Emden and R.A. Kowalski. The Semantics of Predicate Logic as a Programming Language. *J.ACM*, 23(4):733–742, 1976.

[87] A. Van Gelder. Negation as Failure Using Tight Derivations for General Logic Programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 1149–176. Morgan Kaufmann, 1988.

[88] A. Van Gelder, K. Ross, and J.S. Schlipf. Unfounded Sets and Well-founded Semantics for General Logic Programs. In *Proc. 7th Symposium on Principles of Database Systems*, pages 221–230, 1988.

[89] A. Yahya and L.J. Henschen. Deduction in Non-Horn Databases. *J. Automated Reasoning*, 1(2):141–160, 1985.

4 Scientific personnel and degrees completed with ARO support during previous grant period from July 1988 to June 1991:

Also in the same time period, under support from the ARO grant a number of graduate degrees were awarded, as indicated in the following list of personnel supported.

- 5 MS degrees
- 6 Ph.D. degrees

Dr. Jack Minker (PI)
Dr. Donald Perlis (PI)
Dr. Irene Durand (Post Doctoral Visitor)
Dr. Jennifer Elgot-Drapkin (Ph.D. 1988)
Dr. Annie Gal, (Ph.D. 1988), University of Rennes, France
(with high honors; directed by Prof. Jack Minker)
Dr. Arcot Rajasekar (Ph.D. 1989)
Dr. Mark Giuliano (Ph.D. 1990)
Dr. Diana Gordon (Ph.D. 1990)
Dr. Jorge Lobo (Ph.D. 1990)
Ms. Teresa Gaasterland (M.S. 1988)
Mr. Thomas Melton (M.S. 1988)
Mr. Michael Miller (M.S. 1988)
Mr. Sanjoy Paul (M.S. 1988)
Mr. Will Sterbenz (M.S. 1989)